Enterprise Integration
EAI vs. SOA vs. ESB

Author - Anurag Goel

TABLE OF CONTENTS

1 INTRODUCTION

This paper aims to bring clarity to terms EAI, ESB, SOA and provide a clear distinction. Also this paper would discuss architecture options available for enterprise integration and what these options are most suitable for. Finally this paper would address the discussions we have seen in various forums about whether EAI is going to be replaced by SOA or ESB.

2 SOA

Service oriented architecture is approach to have software resources in an enterprise available and discoverable on network as well defined services. Each service would achieve a predefined business objective and perform discrete units of work. The services are independent and do not depend on the context or state of the other services. They work within distributed systems architecture.

Earlier SOA used COM or ORB based on CORBA specifications and recent SOA stress on web services using standard description (WSDL), discovery (UDDI) and messaging (SOAP). Service oriented architecture may or may not use web services but yes web services provide a simple way towards service oriented architecture albeit with the age old security and reliability limitations.

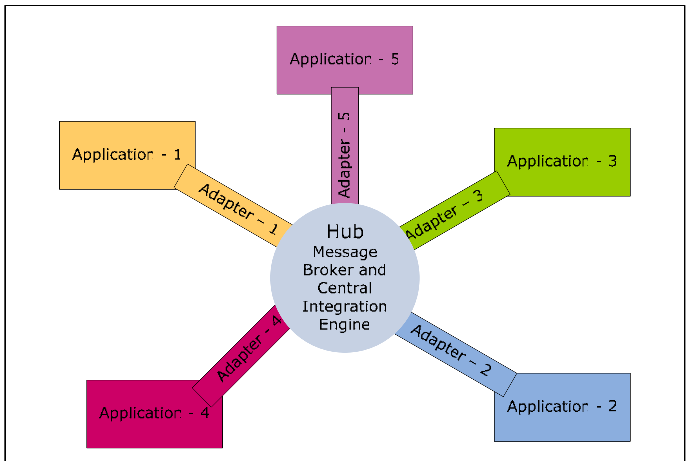
3 EAI

Enterprise application integration is a business need to make diverse applications in an enterprise including partner systems to communicate to each other to achieve a business objective in a seamless reliable fashion irrespective of platform and geographical location of these applications. It is a business need and business never dies it only evolves. I have seen people saying that EAI is a thing of past now SOA is here, it is just like saying "transportation is a thing of past now road is here".

EAI comprises of message acceptance, transformation, translation, routing, message delivery and business process management. Usually messages transportation is asynchronous but for a business need it can be synchronous as well. There are two basic architectures to achieve this, bus and hub/spoke architecture. Both of these can be used to develop services and then it also becomes service orientated architecture.

3.1 HUB / SPOKE

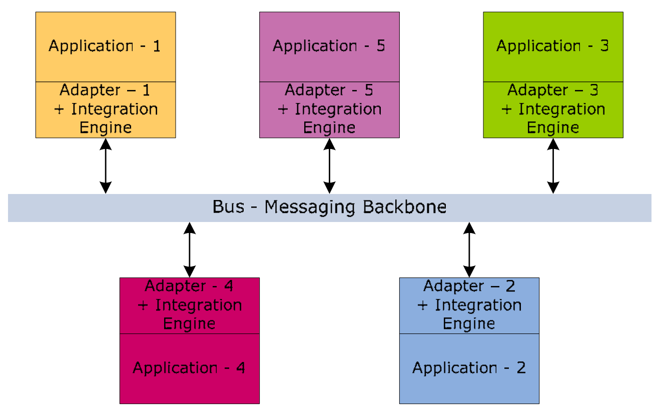
Hub/Spoke architecture uses a centralized broker (Hub) and adapters (Spoke) which connect applications to Hub. Spoke connect to application and convert application data format to a format which Hub understands and vice versa. Hub on the other hand brokers all messages and takes care of content transformation/translation of the incoming message into a format the destination system understands and routing the message. Adapters take data from source application and publish messages to the message broker, which, in turn, does transformation/translation/routing and passes messages to subscribing adapter which sends it to destination application(s). Having a single Hub makes system with this architecture easy to manage but scalability takes a hit. At some point of time as number of messages increase, scalability gets dependent on hardware. Having a bigger box to scale application has never been an ideal solution so to overcome this limitation most vendors have incorporated the concept of federated hub and spoke architecture in which multiple hubs can be present, each hub would have local metadata and rules as well as global metadata. Changes to global rules and metadata are automatically propagated to other hubs. Federated hub spoke architecture alleviates scalability issue while central management of multiple hubs makes this architecture easy to manage and brings down support cost.



Hub/Spoke Architecture: Central Message broker with integration engine and adapters as spoke

3.2 BUS

Bus architecture uses a central messaging backbone (bus) for message propagation. Applications would publish messages to bus using adapters. These messages would flow to subscribing applications using message bus. Subscribing applications will have adapters which would take message from bus and transform the message into a format required for the application. Key difference between hub/spoke and bus topology is that for the bus architecture, the integration engine that performs message transformation and routing is distributed in the application adapters and bus architecture requires an application adapter to run on the same platform as the original applications.
Since adapters have integration engine and run on same platform on which source and target applications run, this scales much better and is complex to maintain compared to hub/spoke topology.


Application - 1
Adapter – 1 + Integration Engine
Application - 5
Adapter - 5 + Integration Engine
Application - 3
Adapter – 3 + Integration Engine
Bus - Messaging Backbone
Adapter - 4 + Integration Engine
Application - 4
Adapter – 2 + Integration Engine
Application - 2

Bus Architecture: Central messaging backbone and distributed integration engine at adapters

4 ESB

Enterprise service bus is an infrastructure to facilitate SOA. It gives API which can be used to develop services and makes services interact with each other reliably. Technically ESB is a messaging backbone which does protocol conversion, message format transformation, routing, accept and deliver messages from various services and application which are linked to ESB.
Current EAI landscape is seeing many vendors who offer enterprise service bus and claim it to be a brand new concept. This brings a question on what exactly is the difference between ESB and the bus based implementations which have been there in market for quite a long time now. Actually there is not much difference between ESB and proprietary buses except for a few subtle ones. Main difference between ESB and proprietary bus implementation is of cost which is significantly low for ESB. Reason for this cost difference is two fold, first proprietary bus offers lot of built in functionalities as a suit of product which need to be developed for ESB implementations based on business requirement, second most proprietary buses use some proprietary formats to enhance the performance and that increases the cost. ESB on the other hand is usually standard based, so it is a tradeoff between performance and cost between proprietary bus and ESB. Main advantage of ESB is that it costs much less then hub/spoke or bus based product suits and that it is standard based.

5 CONCLUSION

Following table give a quick comparison of hub/spoke, bus based product suits and ESB. Also all these three architectures can be service oriented depending on implementation which is reflected in this comparison.

Evaluation Parameter	**Hub Architecture**	**Bus Architecture**	
		Proprietary bus based product suit	**ESB**
Installation Effort	Less installation effort compared to solutions with bus architecture.	Moderate effort	Moderate effort
Administration	Easy to maintain and administrate because of central hub.	Administration may be complex depending upon the integrated systems	Administration may be complex depending upon the integrated systems
Cost	High	High	Low cost because it does not use proprietary formats to enhance performance. Also it does not provide all the services usually provided by proprietary product suits.

Scalability	High if federated architecture is used otherwise limited by the hardware of box used to host Hub	Highly scalable	Highly scalable
Standards	Mostly standard based but may use proprietary internal formats	Mostly standard based but may use proprietary internal formats	Standard based
SOA	Can be implemented as service oriented	Can be implemented as service oriented	Service oriented

SOA brings cost effective, reusable and low lead time solutions to an organization but EAI and SOA are both going to coexist. Web services alone as SOA can not handle the complex, secure and SLA based applications of an enterprise currently and unless we see a technological break through it is going to remain that way.

Enterprise service bus would enable low cost integration and would be used by companies with limited IT resources and environments that involve a handful of systems and moderate transaction volumes. Packaged EAI solutions would have SOA as basic tenet and would continue to be used for large scale integration by companies having huge number of diverse system and high transaction volumes. Next generation EAI solutions would use more and more of SOA to provide reliable, secure, low cost and flexible solutions.

6 TAKEAWAYS SUMMARY

1. SOA brings cost effective, reusable and low lead time solutions to an organization but EAI and SOA are both going to coexist.
2. SOA is more then web services, in fact web services alone can not handle the complex, secure and SLA based applications of an enterprise.
3. Enterprise service bus would enable low cost integration and would be used by companies with limited IT resources
4. Packaged EAI solutions in future would have SOA as basic tenet and would continue to be the prime choice for large scale integration.